

Multi-resource fairness: Objectives, algorithms and performance

Thomas Bonald¹ and James Roberts^{*2}

¹Télécom ParisTech, Paris, France

²IRT System-X, Paris-Saclay, France

October 6, 2014

Abstract

Designing efficient and fair algorithms for sharing multiple resources between heterogeneous demands is becoming increasingly important. Applications include compute clusters shared by multi-task jobs and routers equipped with middleboxes shared by flows of different types. We show that the currently preferred objective of Dominant Resource Fairness has a significantly less favorable efficiency-fairness tradeoff than alternatives like Proportional Fairness and our proposal, Bottleneck Max Fairness. In addition to other desirable properties, these objectives are equally strategyproof in any realistic scenario with dynamic demand.

1 Introduction

Multi-resource fairness has recently received a lot of attention thanks mainly to two papers by A. Ghodsi and colleagues from UC Berkeley [5, 4]. In the present paper we argue that the objective of dominant resource fairness (DRF), advocated by Ghodsi *et al.* in [5] and [4] for resource sharing in compute clusters and routers equipped with middleboxes, respectively, is misguided. More classical sharing objectives, like proportional fairness (PF), achieve a better efficiency-fairness tradeoff. Such objectives were discarded in the UC Berkeley work as being vulnerable to manipulation by users seeking to gain more than their fair share by falsely stating their requirements. However, we argue that the advocated *strategyproofness* property, possessed by DRF but not PF in a context of *static* demand, is not in fact discriminating when considering the more realistic context of *dynamic* demand.

We have already made the case in a recent paper for proportional fair sharing of compute cluster resources [2]. Here we extend those arguments to the case of router resources and introduce a new allocation objective called bottleneck max fairness (BMF). We propose packet-based algorithms to realize PF and BMF that have comparable complexity to the DRFQ algorithms proposed to realize DRF [4].

Multi-resource sharing in a compute cluster consists in launching appropriate numbers of tasks of multi-task jobs. Each task of a given job has its particular requirements for CPU, RAM and other resources. In practice, various constraints on the placement of tasks on physical machines need to be taken into account but, for present purposes, we follow Ghodsi *et al.* [5] and assume resources of the same type are assembled in homogeneous pools. The question is, how should a central scheduler determine the numbers of simultaneous tasks to run for all the currently active jobs, ensuring efficiency and some measure of fairness.

Routers increasingly employ middleboxes to process packets and these constitute potential bottlenecks for flows, in addition to link bandwidth. Different flows have different per-packet resource requirements (e.g., some require complex packet processing, others none) and the issue here is what packet rates should be imposed for concurrent flows in order to efficiently and fairly share all types of resource.

Bandwidth sharing in a network is a particular form of multi-resource sharing. The problem is generally simpler since all flows sharing a given link are assumed to have identical requirements. This assumption is not true, however, for some wireless links where the resource to be allocated is spectrum

^{*}The authors are members of the LINCS, Paris, France. See www.lincs.fr.

time and the amount required to realize a given bit rate varies considerably between terminals, depending on their particular radio conditions. It is important to share wireless and wired links in a manner that adequately balances efficiency and fairness.

We first describe DRF, PF and BMF sharing objectives in abstract terms common to the above three application contexts. We define the corresponding allocations and discuss their significant characteristics. Algorithms are then presented for realizing the respective allocations. Our main original contribution here is to identify practical, packet-based algorithms that realize PF and BMF sharing between flows in a router or network. The following section on performance presents numerical results that justify our claim that DRF is not in fact the best objective. Under dynamic demand, where jobs or flows arrive over time and have finite size, completion times are consistently and significantly smaller with either PF or BMF allocations.

2 Objectives

We define DRF, PF and BMF multi-resource sharing objectives with respect to a fluid model where resources are assumed infinitely divisible. The model is abstract and, rather than jobs or flows, we refer to *transactions*. These are assumed divisible into a large number of infinitesimal components having homogeneous resource usage characteristics.

2.1 A fluid model

Consider J infinitely divisible resources of normalized capacity 1 to be shared by n transactions indexed by i . Each transaction i requires amounts of each resource in fixed proportions and the amount allocated determines the rate at which the transaction progresses. We denote the requirements of transaction i by a vector $a_i = (a_{i1}, \dots, a_{iJ})$ where, for definiteness, and without loss of generality, we set $\max_j(a_{ij}) = 1$. A resource for which $a_{ij} = 1$ is called a *dominant resource* for transaction i .

An allocation is defined by a vector of real numbers $\varphi = (\varphi_1, \dots, \varphi_n)$ such that $\varphi_i a_{ij}$ is the fraction of resource j allocated to transaction i . The allocation must satisfy capacity constraints:

$$\sum_{i=1}^n \varphi_i a_{ij} \leq 1, \text{ for } j = 1, \dots, J. \quad (1)$$

Following Ghodsi *et al.* [5], allocations should have the following properties. They should be *Pareto efficient* in that no fraction of any resource should be left needlessly idle. They should offer a *sharing-incentive* in that any transaction is assured at least a fraction $1/n$ of its dominant resource. They should be *single resource fair* in the sense that $\varphi_i a_{i1} = 1/n$ when $J = 1$. It may readily be verified that the following three allocations have these properties [5, 2, 3].

2.2 Dominant resource fairness

Dominant resource fairness (DRF) is the unique Pareto efficient allocation where transactions obtain fractions of their dominant resource that are as equal as possible. Equality may not be possible if some a_{ij} are zero [14]. The main advantage of DRF identified in [5] is that, in addition to the above-mentioned properties, it is *strategyproof*. This means a transaction cannot gain a greater share of resources by boosting some component of its requirement vector.

2.3 Proportional fairness

An allocation φ is proportional fair (PF) if, for any other allocation ϕ satisfying (1), the sum of proportional changes is negative or zero, $\sum_i (\phi_i - \varphi_i) / \varphi_i \leq 0$ [11]. Equivalently, φ maximizes $\sum_i \log \varphi_i$ subject to (1) and can thus be said to maximize social welfare assuming a logarithmic utility function. For a single resource, the allocation is such that the $\varphi_i a_{i1}$ are equal, fulfilling single resource fairness. It can be shown that PF is the only utility-based allocation with this property [2]. The PF allocation is unique.

While PF was introduced in [11] as an objective for sharing bandwidth in a wired network, it was advocated independently by Tse and co-authors as the preferred allocation for a time-shared wireless

downlink channel [16]. Here $1/a_{i1}$ represents the number of bits transmissible per constant length time slot and depends on the radio conditions of receiver i . Allocations φ_i are measured in bit/s and the PF allocation is such that each transaction receives the same slot rate $\varphi_i a_{i1}$.

PF is *not* strategyproof: if a user knows the requirements of concurrent transactions, it can obtain a bigger allocation by judiciously increasing its requirement for some non-dominant resource. For example, consider $n = 2$ jobs and $J = 2$ resources with $a_1 = (1/2, 1)$ and $a_2 = (1, 1/2)$. The PF allocation is then $\varphi = (2/3, 2/3)$. If transaction 1 claims $2/3$ of resource 1 instead of $1/2$, we find $\varphi' = (3/4, 1/2)$ yielding a bigger share for transaction 1 at the expense of transaction 2.

Note, however, that if a user does not know the other requirements, an ill-chosen modification can instead lead to it receiving a smaller share. Continuing the above example, if transaction 1 claims $a_1 = (1, 1)$, the PF allocation would be $\varphi' = (1/2, 1/2)$. Both transactions lose out compared to the result of the truthful declaration, $a_1 = (1/2, 1)$.

2.4 Bottleneck max fairness

Dolev *et al.* proposed the “no justified complaints” objective as an alternative to DRF [3]. To be concise, we adopt a simplified definition of this objective: a transaction has no justified complaint if it receives at least a fraction $1/n$ of at least one fully allocated resource. The latter constitutes a bottleneck and Pareto efficient allocations fulfilling this objective for all users have been said to realize “bottleneck-based fairness” [18, 8].

Unlike DRF and PF, there is in general no unique bottleneck-based fair allocation according to the above definition. We restrict the class of such allocations somewhat by requiring in addition that every transaction i receives an allocation $\varphi_i a_{ij}$ of some bottleneck resource j that is *maximal* for that resource. We call such allocations “bottleneck max fair” (BMF).

It can be shown that a system with 2 resources has a unique BMF allocation but that, for 3 or more resources, there can be a continuum of allocations realizing the BMF sharing objective. For example, consider a 3 resource system with 3 transactions having requirement vectors $(1, 1, 1)$, $(1, 1/2, 3/4)$ and $(1/2, 1, 3/4)$. Allocations $\varphi^{(0)} = (2/5, 2/5, 2/5)$ and $\varphi^{(1)} = (1/3, 4/9, 4/9)$ both satisfy the definition. In fact, allocations $\varphi^{(x)} = (2/5 - x/15, 2/5 + 2x/45, 2/5 + 2x/45)$ for $0 \leq x \leq 1$ are all BMF.

It can be shown that BMF coincides with PF for $n = 2$ and $J = 2$ and is therefore not strategyproof.

3 Algorithms

We discuss algorithms to realize DRF, PF and BMF, successively for the fluid model, for a compute cluster shared by multi-task jobs and for router or network resources shared by flows of packets.

3.1 Dominant resource fairness

To realize the ideal DRF allocation of Section 2.2 one can employ a water-filling algorithm [5, 14]: with the dominant resource requirement normalized to 1, the φ_i are increased at the same rate until some resource is fully used; transactions using that resource are frozen while rates of the others with non-zero requirements on non-saturated resources are increased together until a second resource is full; the process continues until all the φ_i are frozen.

Ghodsi *et al.* [5] also show how the DRF allocation can be realized in practice for compute cluster resources. As resources are freed on task completion they are re-allocated preferentially to the most deprived job. This is the job for which the difference between the ideal and current allocations of its dominant resource is greatest.

To share router resources between flows, we consider the simpler case where all packets of the same flow have the same requirement vector. The memoryless DRFQ algorithm defined by Ghodsi *et al.* [4] then applies. We suppose there is a fixed-size window of W packets for each flow and that buffers are sized for no loss. This window might be realized within a router by creating an ingress queue and only admitting packet k when packet $k - W$ has finished processing at all resources.

Memoryless DRFQ determines the order in which packets are served at each resource through an adaptation of start-time fair queuing (SFQ) [7]. The virtual start time S_i^k of packet k of flow i is

determined recursively,

$$S_i^k = \max \left(V(u_i^k), S_i^{k-1} + \max_j \{a_{ij}\} \right), \quad (2)$$

where u_i^k is the packet arrival time and the virtual time function of real time t , $V(t)$, is set equal to the largest start time at t of any packet to have begun service at any resource. Packets are served at each resource in increasing order of the S_i^k . Note that, with our normalized requirements, the max in (2) is identically equal to 1.

This algorithm is generalized in [4], as “dovetailing DRFQ”, to account for successive packets of the same flow having different requirements.

3.2 Proportional fairness

The PF allocation for the fluid model can be derived on applying the Karuch-Kuhn-Tucker theorem to maximize $\sum \log(\varphi_i)$ subject to capacity constraints (1). We must find Lagrange multipliers ν_j satisfying

$$\frac{1}{\varphi_i} = \sum_{j=1}^J a_{ij} \nu_j, \quad (3)$$

for $i = 1, \dots, n$, where $\nu_j \geq 0$ and $\nu_j > 0$ if and only if $\sum_{i=1}^n \varphi_i a_{ij} = 1$.

The optimal solution can be used in a practical task-based algorithm for sharing a compute cluster. This consists in applying the same “most deprived job” approach from Section 3.1 with, of course, an alternative criterion to measure deprivation [2].

An algorithm for packet-based PF can be derived on adapting the analysis of Massoulié and Roberts for network bandwidth sharing [13]. We denote the number of flow i packets waiting or in service at resource j by Q_{ij} and let $Q(j) = \sum_i Q_{ij}$. Assuming as above that each flow maintains a fixed window of W packets, we have the conservation equation,

$$W = \sum_j Q_{ij}. \quad (4)$$

We assume the system attains a stable regime where the Q_{ij} are positive constants when resource j is a bottleneck, or zero otherwise. It turns out that serving packets at rates proportional to Q_{ij}/a_{ij} yields the PF allocation. We have,

$$\sum_i \varphi_i a_{ij} = 1 \Rightarrow \frac{Q_{i'j}}{Q_{ij}} = \frac{\varphi_{i'} a_{i'j}}{\varphi_i a_{ij}}$$

and summing over i' yields $Q_{ij} = \varphi_i a_{ij} Q(j)$. On substituting for Q_{ij} in (4) we derive,

$$\frac{1}{\varphi_i} = \sum_j a_{ij} \frac{Q(j)}{W}$$

where $Q(j) \geq 0$ and $Q(j) > 0$ if and only if $\sum_i \varphi_i a_{ij} = 1$. Comparison with (3) shows that the $Q(j)/W$ coincide with the Lagrange multipliers ν_j and, therefore, that φ is indeed the unique PF allocation.

To realize the required packet rates we can adapt the SFQ algorithm. Start times S_{ij}^k are defined recursively and independently for each resource j ,

$$S_{ij}^k = \max \left(V_j(u_{ij}^k), S_{ij}^{k-1} + a_{ij}/Q_{ij} \right), \quad (5)$$

where u_{ij}^k is the packet arrival time at resource j and virtual time $V_j(t)$ is the start time at t of the last packet to have begun service at j .

This algorithm can be applied when the a_{ij} vary from packet to packet realizing a dovetailing PF.

3.3 Bottleneck max fairness

For small systems, it is possible to determine a BMF allocation by testing the feasibility of all possible one-to-one mappings of transactions to potential bottlenecks. For a mapping to be feasible, it must be

possible to find values φ_i satisfying (1) such that every resource j with at least one mapped transaction is a bottleneck ($\sum \varphi_i a_{ij} = 1$) and the allocations of transactions mapped to j are equal (i.e., $\varphi_i a_{ij} = \varphi_{i'} a_{i'j}$ if i and i' are mapped to j).

For a system limited to two resources, it can be shown that this procedure yields the unique BMF allocation. However, to prove the existence of a BFM allocation in general is challenging (see [3] and [8]) though the following practical algorithms suggest this is true.

The most deprived job approach can be adapted to realize BMF in a cluster. For every job i we note its rank at each resource j : jobs with the biggest allocation have rank 1, jobs of rank k for $k > 1$ have a smaller share than $k - 1$ other jobs. The deprivation status of job i is its minimum rank on a bottleneck resource. Tasks are launched preferentially for the job whose current status is largest.

For the router application, BMF can be realized by imposing local weighted max-min fairness at each resource with respective weights $1/a_{ij}$. To demonstrate this, we again assume the existence of a steady state and adapt arguments from [13]. We assume each flow i maintains a window of W packets and, to include the possibility of remotely located resources (as in the network application), we introduce a round trip propagation time T_i . The following conservation relations generalize (4),

$$W = \varphi_i T_i + \sum_j Q_{ij}. \quad (6)$$

Relation (6) shows that for every transaction i , as long as $W > \varphi_i T_i$, there is at least one bottleneck resource j such that $Q_{ij} > 0$ and $\sum_i \varphi_i a_{ij} = 1$. Moreover, the weighted fair queuing scheduler at j ensures $\varphi_i a_{ij} = \max_{i'} (\varphi_{i'} a_{i'j})$, completing the BMF defining conditions.

To realize BMF using SFQ, start times must be calculated as follows,

$$S_{ij}^k = \max(V_j(u_i^k), S_{ij}^{k-1} + a_{ij}), \quad (7)$$

where u_i^k and $V_j(t)$ are as defined in Section 3.2.

As for DRF and PF, this algorithm does not need the a_{ij} of successive packets to be constant, leading to a corresponding dovetailing algorithm.

4 Performance

While the sharing algorithms are defined with respect to a fixed set of transactions, their performance can only be realistically appraised under dynamic demand where transactions occur over time, each bringing a finite amount of work to be accomplished. We propose a simple Markovian demand model and use it to compare the completion time performance of DRF, PF and BMF.

4.1 Markovian demand model

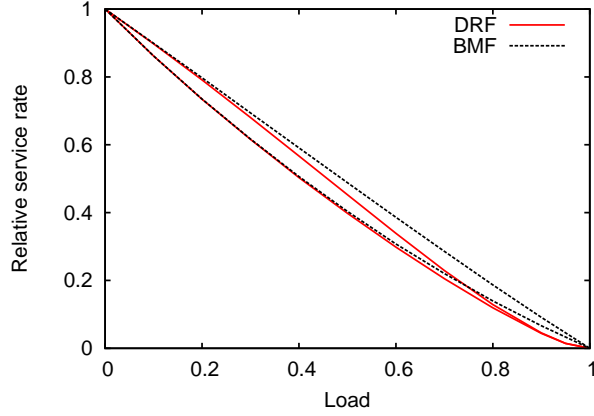
We suppose transactions belong to one of K classes and transactions of class k arrive as a Poisson process of rate λ_k . Class k transactions have requirement vector (a_{k1}, \dots, a_{kJ}) and bring an exponentially distributed amount of work (number of tasks \times mean task duration or size of a flow in bytes, say) of mean $1/\mu_k$. The state vector (n_1, \dots, n_K) , giving the current number of transactions in progress, is then a Markov process with component- k birth rate λ_k and death rate $n_k \varphi_k \mu_k$.

This process is stable as long as loads $\rho_k = \lambda_k / \mu_k$ satisfy the following inequalities,

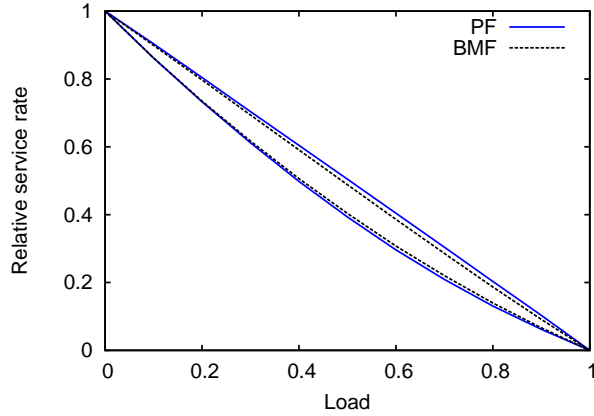
$$\sum_{k=1}^K \rho_k a_{kj} < 1, \quad (8)$$

for $j = 1, \dots, J$. In this case, the process has a stationary distribution $\pi(n)$ from which we can compute performance measures like expected completion times.

In the following we compare algorithm performance via the mean service rate γ_k , defined as the ratio of the mean work $1/\mu_k$ to the mean completion time. Its reciprocal is thus a normalized completion time. Using Little's law, we find $\gamma_k = \lambda_k / \mathbb{E}(n_k)$ where $\mathbb{E}(n_k)$ is the mean number of class k transactions in progress. See [1] for more information on this model including a discussion on the non-criticality of Poisson and exponential assumptions.



(a) BMF v DRF



(b) BMF v PF

Figure 1: Service rates γ_k against resource load for BMF and DRF with balanced load: $a_1 = (.1, 1)$, $a_2 = (1, .1)$, $a_3 = (1, 1)$; $\rho_1 = \rho_2 = \rho_3$; for each allocation we have, $\gamma_1 = \gamma_2 > \gamma_3$.

4.2 Strategyproofness

First reconsider the notion of strategyproofness in the context of dynamic demand. While it may be possible for an agent to know the requirement vectors of competing classes, it is hardly reasonable to suppose knowledge of the current numbers of transactions in progress. Moreover, these numbers change rapidly and a good strategy in one state may rapidly become disadvantageous as the state changes.

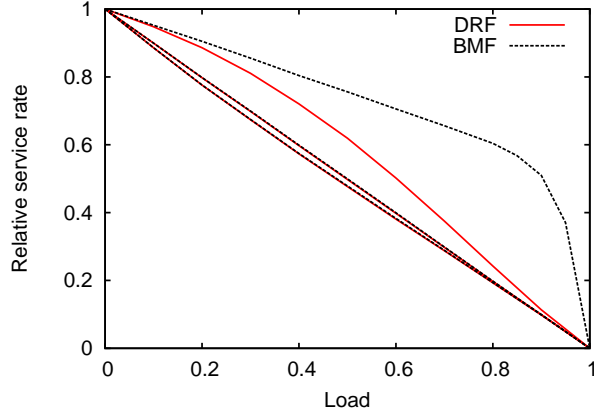
For instance, PF was shown in Section 2.3 to be vulnerable to a falsely declared requirement vector: transaction 1 with requirement vector $a_1 = (1/2, 1)$ competing with transaction 2 with vector $(1, 1/2)$ was shown to gain from a falsely declared vector $(2/3, 1)$. If now, there are 2 transactions like the second, the same strategy results in transaction 1 receiving $\varphi_1 = 1/2$, less than the $2/3$ obtained with a true declaration.

We have identified no winning strategy for any of the considered allocations and, in the absence of any counter-examples, consider them all to be equally strategyproof. Note, in particular, that an increase in any requirement a_{ij} can diminish the system capacity region (8) degrading performance for all.

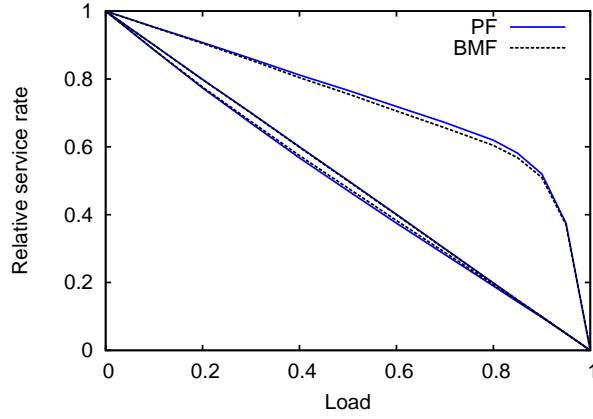
4.3 Performance of the fluid model

We illustrate the relative performance of DRF, PF and BMF using a numerical example. Two resources are shared by 3 classes of transaction with requirement vectors $a_1 = (.1, 1)$, $a_2 = (1, .1)$, $a_3 = (1, 1)$. The figures plot realized service rates γ_k against the load of the heaviest loaded resource ($\text{argmax}_j (\sum_i \rho_i a_{ij})$).

Figure 1 corresponds to balanced load $\rho_1 = \rho_2 = \rho_3$ and shows performance is similar for all three



(a) BMF v DRF



(b) BMF v PF

Figure 2: Service rates γ_k against resource 2 load for BMF and DRF with unbalanced load: $a_1 = (.1, 1)$, $a_2 = (1, .1)$, $a_3 = (1, 1)$; $\rho_1 = 4\rho_2 = 4\rho_3$; for each allocation we have, $\gamma_2 > \gamma_1 > \gamma_3$.

allocations. BMF is very close to PF while both are somewhat better than DRF, especially at high load.

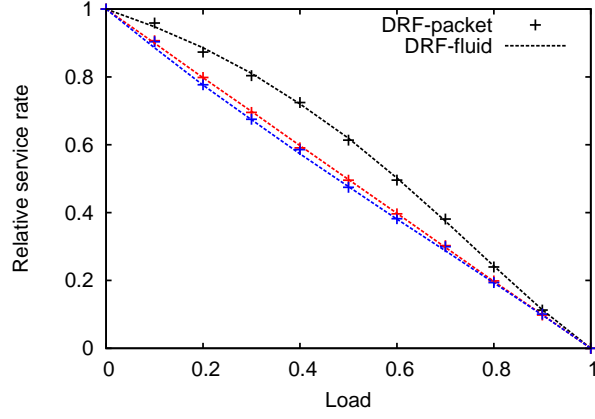
The difference in performance is accentuated under unbalanced load, as illustrated in Figure 2 for the case $\rho_1 = 4\rho_2 = 4\rho_3$. In this scenario resource 1 has less than half the load of resource 2. The plots show that strict fairness imposed by DRF prevents class 2 transactions from fully exploiting this. The service rates of PF and BMF are roughly equivalent, both yielding a better efficiency-fairness tradeoff with a significant gain in γ_2 for negligible reductions in γ_1 and γ_3 .

4.4 Packet-based allocations

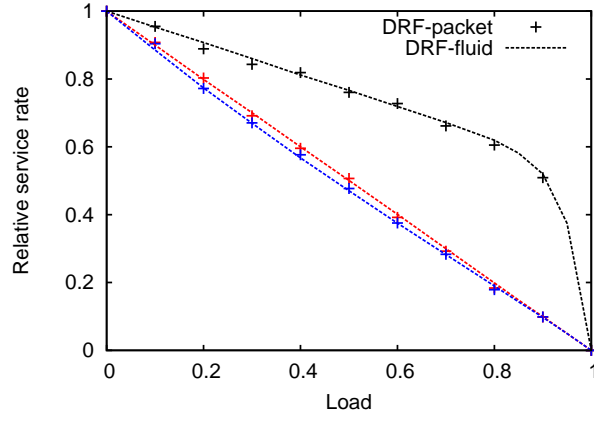
It was shown in [2] that the serve-the-most-deprived-job algorithms yield service rates that retain the same comparative behaviour of DRF and PF illustrated in Section 4.3. In this section we evaluate the effectiveness of the SFQ-based algorithms defined in Section 3 for router resource sharing. We assume the number of packets in each flow has a geometric distribution of mean 20000 and the window W is set to 30 packets. These parameter choices do not critically impact the presented results.

Service rates are shown in Figure 3 for unbalanced load where crosses are results of packet-based simulations for 10^5 flow arrivals and lines are the fluid model results from the previous subsection. The results confirm that all three packet-based algorithms closely approximate the service rates of the ideal allocations.

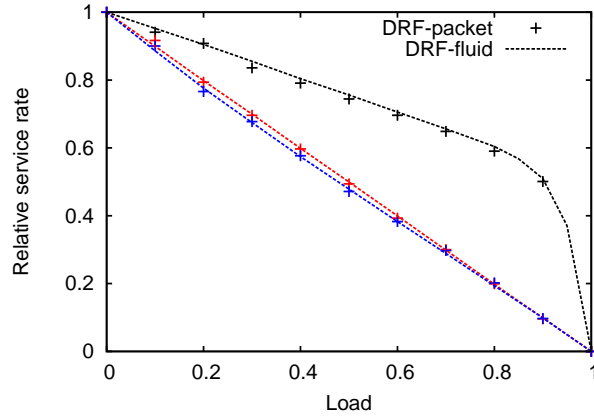
Given the similarity of the results for PF and BMF, it would be preferable to implement the algorithm for the latter as it is simpler in not requiring knowledge of per-flow queue lengths. Moreover, BMF is the



(a) DRF



(b) PF



(c) BMF

Figure 3: Service rates γ_k against resource 2 load for unbalanced load: $a_1 = (.1, 1)$, $a_2 = (1, .1)$, $a_3 = (1, 1)$; $\rho_1 = 4\rho_2 = 4\rho_3$; for each allocation, $\gamma_2 > \gamma_1 > \gamma_3$.

only algorithm applicable when the propagation time is non-negligible, as in the case of a wireless access network.

5 Related work

We limit the present discussion to the most relevant related work. DRF [5] and “no justified complaints” [3] were placed in a more general economics framework in the work of Gutman and Nisan [8]. Joe-Wang *et al.* also generalized DRF by introducing two families of allocations that allow a controlled tradeoff between efficiency and fairness [10]. All these objectives are evaluated assuming a fixed set of transactions.

The “serve the most deprived job” approach introduced in [5] proves very versatile. It is used by Zeldes and Feitelson [18] to implement bottleneck-based fairness and by Ghodsi and co-authors [6] to account for compatibility constraints in task placement. Our proposed implementations of PF and BMF for sharing cluster resources are further illustrations of this versatility.

The packet-based algorithms designed by Ghodsi *et al.* [4] to realize DRFQ for shared router resources are based on start-time fair queuing. Wang and co-authors have proposed alternative realizations that adapt DRR [15] to the multi-resource context [17]. Our implementations of PF and BMF rely on SFQ though it appears straightforward to substitute alternative fair queuing algorithms if required.

The need to evaluate the performance of resource sharing objectives under dynamic demand is still not widely recognized. The paper by Massoulié and Roberts [12] was perhaps the first to note the importance of this while some of the most significant subsequent findings are summarized in Bonald *et al.* [1]. Our earlier paper [2] and the present work extend this analysis to the domain of multi-resource sharing.

6 Conclusions

Multi-resource sharing for efficiency and fairness is an old issue in networking with challenging new variants occurring in the domains of cluster computing and software routers. Recent prominent publications have led to the emergence of DRF and its apparent acceptance as the preferred sharing objective¹.

We have argued in this paper that this popularity is misplaced since alternative objectives like PF display a better efficiency-fairness tradeoff. This result is revealed on considering the completion time performance of alternative objectives under the realistic and crucial assumption that demand is dynamic: jobs and flows occur over time and their completion times depend critically on the implemented resource sharing objective.

We have proposed BMF as a pragmatic alternative to PF. It has similar performance and can be realized more simply, especially when sharing router and network resources between flows. It is clearly the most practical objective for a network integrating radio access and wired backhaul. Independent schedulers simply share their own resource equitably. BMF thus generalizes network-wide max-min fairness that is known to be realized by fair schedulers acting independently on each link [9].

Concerns about the vulnerability of objectives like PF and BMF to malicious gaming have been shown to be unfounded. While users can manipulate allocations by falsely boosting their declared requirement of non-dominant resources, their gain depends critically on knowing the requirements of competitors. Such knowledge is clearly inconceivable in the context of highly dynamic populations of active transactions occurring in a realistic model of demand.

The present work is clearly incomplete. It remains notably to more thoroughly evaluate the proposed algorithms under realistic demand models and accounting for practical resource usage constraints. From the theory point of view, it is necessary to carefully analyse the convergence of proposed algorithms and the stability characteristics of supposed equilibria.

¹DRF is implemented in the Hadoop Next Generation Fair Scheduler, for instance.

References

- [1] T. Bonald, L. Massoulié, A. Proutière, and J. Virtamo. A queueing analysis of max-min fairness, proportional fairness and balanced fairness. *Queueing Syst. Theory Appl.*, 53(1-2):65–84, June 2006.
- [2] T. Bonald and J. Roberts. Enhanced cluster computing performance through proportional fairness. *Performance Evaluation*, 79:134–145, September 2014.
- [3] D. Dolev, D. G. Feitelson, J. Y. Halpern, R. Kupferman, and N. Linial. No justified complaints: On fair sharing of multiple resources. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 68–75, New York, NY, USA, 2012. ACM.
- [4] A. Ghodsi, V. Sekar, M. Zaharia, and I. Stoica. Multi-resource fair queueing for packet processing. In *Proceedings of ACM SIGCOMM 2012*, pages 1–12, New York, NY, USA, 2012. ACM.
- [5] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica. Dominant resource fairness: Fair allocation of multiple resource types. In *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, NSDI'11, pages 24–24, Berkeley, CA, USA, 2011. USENIX Association.
- [6] A. Ghodsi, M. Zaharia, S. Shenker, and I. Stoica. Choosy: Max-min fair sharing for datacenter jobs with constraints. In *Proceedings of the 8th ACM European Conference on Computer Systems*, EuroSys '13, pages 365–378, New York, NY, USA, 2013. ACM.
- [7] P. Goyal, H. Vin, and H. Cheng. Start-time fair queueing: a scheduling algorithm for integrated services packet switching networks. *Networking, IEEE/ACM Transactions on*, 5(5):690–704, Oct 1997.
- [8] A. Gutman and N. Nisan. Fair allocation without trade. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '12, pages 719–728, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems.
- [9] E. Hahne. Round-robin scheduling for max-min fairness in data networks. *Selected Areas in Communications, IEEE Journal on*, 9(7):1024–1039, Sep 1991.
- [10] C. Joe-Wong, S. Sen, T. Lan, and M. Chiang. Multi-resource allocation: Fairness-efficiency tradeoffs in a unifying framework. In *INFOCOM, 2012 Proceedings IEEE*, pages 1206–1214, 2012.
- [11] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan. Rate control for communication networks: Shadow prices, proportional fairness and stability. *The Journal of the Operational Research Society*, 49(3):pp. 237–252, 1998.
- [12] L. Massoulié and J. Roberts. Bandwidth sharing and admission control for elastic traffic. *Telecommunication Systems*, 15(1-2):185–201, 2000.
- [13] L. Massoulié and J. Roberts. Bandwidth sharing: Objectives and algorithms. *IEEE/ACM Trans. Netw.*, 10(3):320–328, June 2002.
- [14] D. C. Parkes, A. D. Procaccia, and N. Shah. Beyond dominant resource fairness: extensions, limitations, and indivisibilities. In *ACM Conference on Electronic Commerce*, pages 808–825. ACM, 2012.
- [15] M. Shreedhar and G. Varghese. Efficient fair queueing using deficit round robin. *SIGCOMM Comput. Commun. Rev.*, 25(4):231–242, Oct. 1995.
- [16] P. Viswanath, D. Tse, and R. Laroia. Opportunistic beamforming using dumb antennas. *Information Theory, IEEE Transactions on*, 48(6):1277–1294, Jun 2002.
- [17] W. Wang, B. Li, and B. Liang. Multi-resource round robin: A low complexity packet scheduler with dominant resource fairness. In *ICNP 2013*, 2013.

- [18] Y. Zeldes and D. G. Feitelson. On-line fair allocations based on bottlenecks and global priorities. In *Proceedings of the 4th ACM/SPEC International Conference on Performance Engineering, ICPE '13*, pages 229–240, New York, NY, USA, 2013. ACM.

